Enfoque UTE, V.9-N.1, Mar.2018, pp. 138 - 148
http://ingenieria.ute.edu.ec/enfoqueute/
e-ISSN: 1390-6542 / p-ISSN: 1390-9363

Recibido (Received): 2017/11/23
Aceptado (Accepted): 2018/03/30

Acometer contra un ERP con Software Libre

(Attacking an ERP with Open Source Software)

Catalina Astudillo[1], Fabián Carvajal[1], Juan Pablo Carvallo[1], Esteban Crespo-Martínez[1], Marcos Orellana[1], Rosalva Vintimilla[1]

Resumen:
La seguridad de la información es una preocupación creciente en empresas y organizaciones, es más alta aun cuando se vincula a plataformas financieras donde existe información sensible. Este artículo resume las técnicas utilizadas en el *pentesting* realizado al *software* ERP desarrollado en APEX 5 por la Universidad del Azuay; para ello se han contemplado seis etapas que sugieren una prueba de penetración: i) la conceptualización, donde se define el alcance de las pruebas a realizar; ii) la preparación del laboratorio, en la cual se identifican algunas de las herramientas que servirán para el inicio de las pruebas de seguridad; iii) la obtención de información, donde se hace el reconocimiento y escaneo de posibles objetivos que posteriormente serán explorados con mayor profundidad para identificar características intrínsecas que puedan ser aprovechadas; iv) el análisis de las vulnerabilidades encontradas en la etapa anterior; v) la explotación de vulnerabilidades; y vi) la posexplotación, etapa que contempla la destrucción de evidencias del ataque y la conservación de la conexión y los accesos logrados para extraer información. Todas estas etapas fueron efectuadas dentro de las instalaciones de la Universidad del Azuay, al considerar el ambiente de desarrollo en el que actualmente se encuentra este *software*.

Palabras clave: *pentesting*; seguridad informática; *hacking*; ERP; APEX.

Abstract:
Information security is a growing concern in companies and organizations, being even higher when linked to financial platforms where sensitive information exists. This article explains the techniques used in the pentesting performed on the ERP software developed in APEX 5 by the University of Azuay. To achieve this goal, six stages has been considered for perform a penetration test: I) Conceptualization, where is defined the scope of the tests to be performed. II) Preparation of the laboratory, which identifies some of the tools used to initiate the safety tests. III) Obtaining of information, where the possible objects are recognized and scanned in greater depth to identify intrinsic characteristics for subsequently exploit them. IV) Analysis of the vulnerabilities found in the previous stage. V) Exploitation of vulnerabilities; and VI) Post-exploitation, a stage that contemplates the destruction of evidence of the attack and the conservation of the connection and the accesses obtained to extract information. All these stages were carried out within the facilities of the “Universidad del Azuay”, considering the development environment in which this software is currently located.

Keywords: Pentesting; IT Security; Hacking; ERP; APEX.

1. Introducción

Las empresas están obligadas a anticiparse a los diversos escenarios de riesgo de información, debido a una vertiginosa evolución de la informática, el constante cambio tecnológico, y el rápido crecimiento de las múltiples transacciones de negocio, en los cuales la información y los dispositivos computacionales están inmersos en escenarios como ataques de *hackers*, usuarios del sistema, amenazas lógicas, y una gran variedad

[1] Universidad del Azuay, Cuenca – Ecuador ({cvastudillo, fabianc, jpcarvallo, ecrespo, marore, rvintimilla} @uazuay.edu.ec).

adicional. Sin embargo, debido a la falta de conocimiento sobre cómo protegerla adecuadamente o a la complejidad exigida por muchas normas internacionales y mejores prácticas de desarrollo, múltiples organizaciones descuidan asegurarla tal como lo determinó Crespo en su estudio para la propuesta de la metodología ECU@Risk (Crespo, 2017).

Una de las técnicas de evaluación que apoyan al análisis de brechas de seguridad es el *pentesting*, análisis que debe ser realizado por una persona con conocimientos técnicos, pero con principios éticos. Esta última característica diferencia a un atacante común, conocido en esta época como un "*hacker* de sombrero negro", que es una persona que irrumpe la seguridad de la organización con el fin de beneficiarse de la información en ella almacenada.

Este trabajo resume la experiencia de las pruebas de penetración aplicadas a un software ERP desarrollado en Oracle APEX 5 por la Universidad del Azuay, al considerar las técnicas de caja negra, caja blanca y caja gris, además de múltiples herramientas basadas en OSS (Open Source Software). La primera etapa del ataque aplicada consistió en un escaneo a ciegas (caja negra), solamente al contar con la dirección IP del servidor que aloja a este software.

El *pentesting* realizado se limita a cinco tipos de análisis, sugeridos por (UNAM-CERT, 2016) y (The OWASP Project, 2017): i) identificación del contexto; ii) identificación de las vulnerabilidades en el código; iii) evaluación cross-site scripting XSS; iv) inyección SQL y v) denegación de servicios DDoS.

Se identificaron servicios y puertos de escucha que posiblemente no serían revelados en una prueba de caja blanca; comprobando que la lista no contenía únicamente al TCP 8080. Además, la información proporcionada para la ejecución de la técnica de caja blanca ha servido para confirmar los resultados obtenidos con la primera.

Para este trabajo se estableció una guía de pruebas de penetración basada en herramientas para este efecto y además se documentaron las pruebas de penetración realizadas al software ERP que se encuentra en etapa de desarrollo en la Universidad del Azuay.

El documento logrado se clasifica de la siguiente manera: i) el método utilizado; ii) la conceptualización del escenario de pruebas; iii) la definición del laboratorio de pruebas; iv) los resultados obtenidos en las diversas fases del *pentesting*; y v) las conclusiones obtenidas en las pruebas de evaluación a una aplicación desarrollada en Oracle APEX 5.

2. Materiales y métodos

Este trabajo se relaciona con el trabajo "Web security in the finance sector", investigación realizada por Viera y Serrao con título que examina la seguridad web en aplicaciones del sector financiero; asimismo está el trabajo de Rina López denominado "Pentesting on web applications using ethical - hacking", en el cual identifica métodos de *pentesting* en aplicaciones web mediante *hacking* ético; finalmente se cita al aporte de Väli y Lopensky [6], quienes contextualizan el procedimiento para implementar un analizador de vulnerabilidades y evaluar aplicaciones desarrolladas con APEX.

Según la experiencia adquirida por los autores previamente citados, para estas pruebas de penetración se incluyeron los métodos de caja negra (black box), de caja blanca (white box) y de caja gris (gray box). De acuerdo con (Caballero, 2015), la prueba de caja negra es la técnica en la cual se realizan ataques desconociendo la estructura interna de la organización, al ser ejecutada solo con el conocimiento de una dirección IP o la URL provista por parte del personal interesado.

Igualmente, la prueba de caja blanca consiste en realizar las pruebas de ataque considerando toda la información proporcionada, incluyendo: diagramas, detalles sobre el *hardware*, *software*, sistemas operativos, *firewalls*, etc. Esto permite al atacante o "tester" ser más objetivo; sin embargo, con esta técnica pueden quedar algunos detalles sueltos

debido a que no se consideran otros elementos que podrían ser descubiertos con la de caja negra (Caballero, 2015).

Finalmente está la técnica de caja gris, que, para Caballero, simula a un ataque realizado por un empleado descontento quien mantiene un nivel de privilegios adecuado y cuenta con permisos de acceso a la red interna. La información obtenida en el ataque permitió identificar servicios y puertos de escucha que posiblemente no serían revelados como resultados de una prueba de caja blanca.

Los resultados logrados con la aplicación de esta última técnica permitieron confirmar los resultados obtenidos con la técnica de caja negra. Con relación a la prueba de caja gris, se puede considerar a las pruebas internas con un usuario que tenga acceso a la aplicación, el mismo que utilizará los privilegios con los cuales cuenta, pero con sentido malicioso.

Como parte de la metodología se ha incluido un estudio descriptivo-cuantitativo, con el cual se ejecutó un experimento de interrelaciones y se compararon los resultados obtenidos por la aplicación de las diversas herramientas propuestas en cada una de las etapas del ataque. Además, se aplicó la metodología explicativa y concluyente, donde se evidenciaron los resultados de la causa y el efecto que tiene la prueba de penetración.

3. Resultados

Los resultados obtenidos que serán descritos en esta sección son consecuencia del proceso de una prueba de evaluación de seguridad, en la cual se han considerado 6 etapas: i) conceptualización, ii) preparación del laboratorio, iii) obtención de información, iv) identificación de vulnerabilidades, v) modelamiento de amenazas y vi) identificación de contramedidas.

Conceptualización

La etapa de conceptualización es relevante, pues en ella se ha identificado el contexto y definido los objetivos de la prueba de penetración. El equipo de desarrollo del software ERP ha respondido a cuestionamientos como ¿qué le preocupa?, ¿qué parte de la infraestructura o del sistema es de la que sospecha vulnerabilidades? ¿existen dispositivos delicados que deban ser considerados mientras se realiza el *pentesting*?; respuestas que han servido de argumento para planificar el laboratorio e instrumental a utilizar en las etapas posteriores.

Tal como lo sugiere (Weidman, 2014), fue importante conocer el ámbito de direcciones IP que serán consideradas en el análisis y vulneración, el horario de pruebas, la información del contacto que monitoreará las actividades constantemente, y, sobre todo, contar con la autorización escrita. Si el objetivo no forma parte de la organización contratante, es importante que este tercero conozca formalmente las pruebas a las cuales será sometido.

Bajo estas premisas, se planteó ejecutar las pruebas al servidor de desarrollo que mantiene la Universidad del Azuay para el proyecto UDA ERP, *hardware* que tiene una dirección IP privada en la red 172.16.x.x, y el aplicativo web publicado en el puerto 80, únicamente dentro de los predios de la institución. Adicional a esto, el equipo de desarrollo ha mencionado que el *software* está desarrollado en la herramienta Oracle APEX.

El laboratorio

El laboratorio y su preparación requieren de la identificación de las herramientas adecuadas para lograr el objetivo de análisis. Se han considerado herramientas OSS (Open Source Software) ya que, según (Stallman, 2004) el *software* libre es un programa que está en libertad de ser ejecutado en el momento que se lo quiera, sea cual sea el propósito, además de que pueda ser ajustado a las necesidades; eso es, acceso libre al

código fuente. Al reflexionar sobre las bondades expuestas por Stallman en cuanto a OSS, se incluye a Kali Linux, un distro que contiene más de 300 herramientas para realizar pruebas de seguridad (Security, 2017).

Otra herramienta considerada es VEGA, la misma que permite el escaneo de vulnerabilidades mediante la ejecución automatizada de *scripts* de análisis de inyección SQL (SQL Injection) o *scripts* cruzados (Cross-Site Scripting), además de convertirse en un proxy para interceptar conexiones SSL entre clientes y servidores. Adicionalmente se ha considerado el uso de Nikto para el análisis de contexto, ya que esta herramienta es un escáner actualizable usado en la detección de vulnerabilidades de servidores web.

Con relación a las pruebas de código fuente, se han agregado al laboratorio OWASP ZAP, APEX SERT, W3AF y Web Developer Plug In. OWASP ZAP es una *suite* que habilita la ejecución de pruebas de seguridad en aplicaciones y servicios WEB; y APEX SERT (Security Evaluation and Recomendation Tool) es una herramienta que sirve para evaluar las aplicaciones desarrolladas específicamente en APEX, y Web Developer Plug In, desarrollada por Chris Pederik, que sirve para la evaluación de un sitio web, que, para este trabajo, será aplicado específicamente en el análisis de *cookies*. W3AF es un framework para ataques y auditorías de aplicaciones web, cuyo objetivo es identificar vulnerabilidades en el código fuente.

Obtención de información

Para el proceso de obtención de información se han considerado cinco escenarios: i) identificación del contexto, ii) identificación de las vulnerabilidades en el código fuente, iii) análisis de vulnerabilidades mediante la técnica Cross-Site Scripting – XSS, iv) análisis de vulnerabilidades mediante la técnica de inyección SQL, y v) pruebas de denegación de servicio o DDoS.

Para la identificación del contexto se contemplaron dos herramientas: NMAP y Sparta, ambas incluidas en la *suite* Kali. Además, los resultados obtenidos con la técnica de caja negra fueron comparados con los obtenidos con las técnicas de caja blanca y caja gris.

Sparta identificó más puertos abiertos que la herramienta NMAP. La *Tabla 1* resume los resultados obtenidos en el análisis:

Tabla 1. Resultados del reconocimiento y escaneo

	Resultados	
Herramienta utilizada	NMAP	Sparta
Puertos encontrados	7 puertos abiertos	17 puertos abiertos, 7 filtrados
Puertos abiertos	22, 25, 1521, 4848, 7676, 8080, 8181	22, 25, 1521, 3700, 3820, 3920, 4848, 7676, 7776, 8080, 8181, 8686, 13335, 17210, 29573
Sistema Operativo	Virtualizado en Oracle Virtual Box	Oracle virtualbox
Servidor web	Glassfish 4.1.1.	Glassfish 4.1.1.
Métodos HTTP aceptados	POST, GET	POST, GET
Acepta solicitudes ICMP	SÍ	SÍ

Identificación de vulnerabilidades

Aparecen a la vista las primeras vulnerabilidades: el nombre de host '172.16.1.87' no concuerda con el nombre del certificado Localhost; no está establecida la cabecera de transporte seguro, no está definida la cabecera de protección contra ataques XSS, cabeceras antisecuestro de clic no establecidas, y, entre otras, el método HTTP POST habilitado. También se pudo ver que el sitio no se encuentra cifrado y que utiliza un certificado digital incoherente.

Otra prueba interesante en el reconocimiento fue la de identificar bugs o posibles fallos que pueden aparecer en *software* que es puesto en producción. Estos bugs no solamente se resumen en código, pues concordando con lo que menciona el proyecto OWASP, las organizaciones, instituciones y empresas en general deberían inspeccionar a detalle cada una de las etapas del ciclo de vida de desarrollo de *software*, en las cuales se deben incluir pruebas para garantizar que la seguridad es adecuadamente cubierta por los controles utilizados a lo largo del proceso de desarrollo; programa de pruebas que debería considerar la inclusión de gente, procesos (políticas y normas) y tecnología (The OWASP Project, 2017).

Con un puerto débil conocido (8080), se aplicó la herramienta OWASP Zed Attack Proxy con el método GET y se identificaron vulnerabilidades en el código como la debilidad de uso de contraseñas simples, o la no validación de números de cédula como 1212121212.

Para complementar el análisis de validación de fortalezas de contraseña, se utilizó la herramienta https://www.grc.com/haystack.htm que está disponible en la web; que según (Spendolini, 2016) visualiza el tiempo que tomaría al atacante vulnerar la misma. De esta manera, una contraseña débil como la encontrada en el sistema, que contiene 8 caracteres alfabéticos, podría ser vulnerada en 2.17 segundos mediante un ataque de fuerza bruta en un equipo que pueda generar un billón de consultas por segundo.

La herramienta W3AF ha permitido identificar vulnerabilidades en el código mediante una técnica llamada *fuzzing*, que, según (Hernández, 2007), consiste en un proceso semiautomático que combina técnicas de prueba como son funcional, caja negra, exploratoria, seguridad y resistencia para descubrir fallos de seguridad. Entre las vulnerabilidades encontradas está una lista de destinos que no cuentan con protección contra ataques *clickjacking*, puntos de inyección XSS y además comentarios en el código, como usuarios, versiones, etc.

Con la herramienta VEGA, se ejecutó un ataque XSS seleccionando la opción XSS Injection checks. Se agregaron opciones de análisis de autenticación sobre protocolos no cifrados, detección de carga de archivos al servidor (debido a que se encontró el método POST).

Con la misma herramienta se realizan ataques de inyección SQL, la aplicación es inmune a este tipo de amenazas.

Entre los ataques realizados al perímetro y al servidor realizados con HPING3, la protección perimetral no pudo detectar ataques del tipo IPSpoofing y ataques de denegación de servicio. Se pudo burlar al *firewall* con el comando hping3 --rand-source 172.16.x.x -p 8080, ya que el parámetro –rand-source permitió generar direcciones aleatorias, haciendo que el IDS y el firewall lo consideren como tráfico normal.

Utilizando el Metasploit Exploit Framework de Kali, se logró burlar el *firewall* y hacer efectiva la denegación de servicio por ataque de inundación o SYNFlood.

Modelamiento de amenazas

Esta etapa hace referencia a la creación de una lista de amenazas, en la cual se deberá incluir el título, una descripción breve, el activo (activos), los impactos, el riesgo, las técnicas de mitigación, el estado de mitigación y número del error. Las posibles amenazas para este escenario de análisis pueden resumirse en la *Tabla 2*.

Tabla 2. Resultados del reconocimiento y escaneo

Vulnerabilidad	Amenaza
Puerto TCP 22 y TCP 25 Abierto	Los puertos 22 y 25 no son cifrados y por lo tanto son propensos a ataques de fuerza bruta y ataques MITM (Man in the middle)
Puerto TCP 8080 Abierto	Remplazo de un recurso específico mediante el método POST
Puerto TCP 8080 Abierto	Eliminación de un recurso específico mediante el método DELETE
Puerto TCP 8080 Abierto	HTTP Method ('Allow' Header) Métodos de riesgo potencial: PUT, DELETE
Puerto TCP 8080 Abierto	Remplazo de un recurso específico mediante el método POST
Puerto TCP 8181 Abierto	Eliminación de un recurso específico mediante el método DELETE
Puerto TCP 8181 Abierto	HTTP Method ('Allow' Header) Métodos de riesgo potencial: PUT, DELETE
La cabecera antisecuestro de clic (anti-clickjacking X-Frame-Options header) no está presente	Ataques clickjacking
El servidor presenta posibilidades de fuga mediante ETags	WebBrowsing Fingerprinting
La cabecera de protección contra ataques Cross Site Scripting XSS no está definida	Ataques XSS
Puerto TCP 1521 Abierto	Envenenamiento TNS (Rocha, 2014)
Puerto TCP 3700 Abierto	Portal of Doom. Usado comúnmente por troyanos (Gallo, 2011)
Puertos TCP 3820, 3920, 4848, 7676, 7776, 8080, 8181, 8686, 13335, 17210, 29573, 31780, 33562 Abiertos	Denegación de servicios
El sitio utiliza SSL y la cabecera de transporte seguro estricto (Strict-Transport-Security HTTP Header) no está definida	Denegación de servicios
La cabecera de opciones de tipo de contenido X-Content-Type-Options no está establecida	Denegación de servicios
El nombre de host '172.16.1.87' no concuerda con el nombre del certificado	Suplantación de identidad
Servidor acepta solicitudes ICMP	Saturación ICMP
Acepta Métodos POST y GET	Inundación HTTP
TCP utiliza un mecanismo de negociación de tres vías	Syn Flood
Servidor acepta solicitudes ICMP	Ping de la muerte
Servidor acepta solicitudes ICMP	Inundación IP
Contraseñas débiles	Fuerza bruta
Contraseñas débiles	Ataque de diccionario
Cookies	Predicción de credenciales

De acuerdo con lo que sugiere (Microsoft, 2017), en este paso se debe valorar el riesgo que puede producirse por cada amenaza, y determinar cómo se actuará ante ellas, una vez identificadas las amenazas del entorno. En un *pentesting*, se analizan las posibles amenazas que pueden ser utilizadas para vulnerar los elementos anteriormente hallados. Se pueden usar algunas categorías de efecto para calcular la exposición al riesgo; por ejemplo: daños potenciales, capacidad de reproducción, aprovechamiento, usuarios afectados y capacidad de descubrimiento.

Mediante el uso de las herramientas que sugiere ECU@Risk para la identificación de amenazas, se puede catalogar a que las anteriormente mencionadas pueden caer dentro

de los siguientes grupos: i) errores no intencionados: errores del administrador; ii) errores no intencionados: errores de configuración; iii) provocado: alteración en la configuración; iv) difusión de software dañino.

Según (Alcides, 2009), el riesgo informático es “un conjunto de normas y procedimientos que son aplicados para salvaguardar un sistema informático, cuya finalidad es garantizar que todos los recursos que conforman el sistema informático sean utilizados para el fin que fueron creados sin ninguna intromisión”

Para la creación y el modelamiento de amenazas, se han utilizado las herramientas sugeridas en la metodología ECU@Risk, las mismas que permitieron valorar los riesgos a base de las amenazas identificadas y valoradas. Al aplicar la matriz de impacto (*Tabla 3*) sugerida por la metodología, se pudo llegar a la tabla de valoración de amenazas (*Tabla 4*), en la que se incluye la vulnerabilidad identificada, la amenaza que puede presentarse y su impacto en términos de confidencialidad, disponibilidad e integridad. La última columna hace referencia al valor absoluto de la amenaza, que en resumen es el valor más alto obtenido de las tres perspectivas. Así también, de acuerdo con lo que sugiere la norma, no todas las perspectivas son afectadas, según el tipo de activo de información y la amenaza identificada.

Tabla 3. Matriz de valoración de impacto de ECU@Risk

Matriz de Impacto	Valoración
E – Extremo	5
A – Alto	4
M – Moderado	3
B – Menor	2
L – Leve	1

Identificación de contramedidas

A base de las vulnerabilidades y amenazas identificadas, se han tenido en cuenta priorizar las que mayor impacto pueden ocasionar a la organización, en este caso, al *software* ERP y la infraestructura que lo soporta; sin descuidar los correctivos que puedan realizarse de una forma rápida, como, por ejemplo, denegar las solicitudes ICMP en el *firewall*.

Debe cifrarse el puerto de escucha del aplicativo (uso de HTTPS), y agregar un certificado digital del tipo Organization SSL, el mismo que permite evaluar la autenticidad del sitio y la organización que lo respalda.

Deben cifrarse las bases de datos. Al habilitar esta opción, APEX utilizará el TDE (Transparent Data Encryption) para cifrar los archivos de datos asociados. TDE podría cifrar todos los archivos de bases de datos que son escritos en el disco, haciéndolos ilegibles a cualquiera que trate de acceder a los mismos (Spendolini, 2016).

Es requerido el uso de contraseñas robustas, tanto en la aplicación ERP como en las diferentes herramientas que permiten la gestión de la infraestructura tecnológica que lo soporta, como es el SSH o la base de datos. Según (Spendolini, 2016), una contraseña similar a “Pa5sw0rD.!”, le tomará al atacante 1.83 billón de siglos vulnerarla en modo *online*, 18.28 siglos en modo *offline* y 1.83 años en un escenario de arreglo de *cracking* masivo.

Tabla 4. Valoración de amenazas

Vulnerabilidad	Amenaza	ID Amenaza	C	D	I	V
Puerto TCP 22 Abierto	El puerto 22 no es cifrado y por lo tanto es propenso a ataques de fuerza bruta y ataques MITM	[NO_INTENCIONADO.4]			4	4
Puerto TCP 25 Abierto	El puerto 25 no es cifrado y por lo tanto es propenso a ataques de fuerza bruta y ataques MITM	[NO_INTENCIONADO.4]			3	3
Puerto TCP 8080 Abierto	Remplazo de un recurso específico mediante el método POST	[PROVOCADO.1]	4	3	4	4
Puerto TCP 8080 Abierto	Eliminación de un recurso específico mediante el método DELETE	[PROVOCADO.1]	3	3	4	4
Puerto TCP 8080 Abierto	HTTP Method ('Allow' Header) Métodos de riesgo potencial: PUT, DELETE	[NO_INTENCIONADO.4]			3	3
Puerto TCP 8181 Abierto	Remplazo de un recurso específico mediante el método POST	[PROVOCADO.1]	4	3	4	4
Puerto TCP 8181 Abierto	Eliminación de un recurso específico mediante el método DELETE	[PROVOCADO.1]	3	3	4	4
Puerto TCP 8181 Abierto	HTTP Method ('Allow' Header) Métodos de riesgo potencial: PUT, DELETE	[NO_INTENCIONADO.4]			3	3
La cabecera antisecuestro de clic (anti-clickjacking X-Frame-Options header) no está presente	Ataques clickjacking	[PROVOCADO.1]	2	2	2	2
El servidor presenta posibilidades de fuga mediante ETags	WebBrowsing Fingerprinting	[PROVOCADO.1]	2	2	2	2
La cabecera de protección contra ataques Cross Site Scripting XSS no está definida	Ataques XSS	[PROVOCADO.1]	3	3	3	3
Puerto TCP 1521 Abierto	Envenenamiento TNS (Rocha, 2014)	[PROVOCADO.1]	3	3	3	3
Puerto TCP 3700 Abierto	Portal of Doom. Usado comúnmente por troyanos (Gallo, 2011)	[EL.1]	3	3	3	3
Puertos TCP 3820, 3920, 4848, 7676, 7776, 8080, 8181, 8686, 13335, 17210, 29573, 31780, 33562 Abiertos	Denegación de servicios	[PROVOCADO.1]	3	3	3	3
El sitio utiliza SSL y la cabecera de transporte seguro estricto (Strict-Transport-Security HTTP Header) no está definida	Denegación de servicios	[PROVOCADO.1]	3	1	3	3
La cabecera de opciones de tipo de contenido X-Content-Type-Options no está establecida	Denegación de servicios	[PROVOCADO.1]	3	1	3	3
El nombre de host '172.16.1.87' no concuerda con el nombre del certificado	Suplantación de identidad	[PROVOCADO.1]	4	1	3	4
Servidor acepta solicitudes ICMP	Saturación ICMP	[PROVOCADO.1]	3	3	3	3
Acepta Métodos POST y GET	Inundación HTTP	[PROVOCADO.1]	3	3	3	3
TCP utiliza un mecanismo de negociación de tres vías	Syn Flood	[PROVOCADO.1]	3	3	3	3
Servidor acepta solicitudes ICMP	Ping de la muerte	[PROVOCADO.1]	1	3	1	3
Servidor acepta solicitudes ICMP	Inundación IP	[PROVOCADO.1]	1	3	1	3
Contraseñas débiles	Fuerza bruta	[PROVOCADO.1]	4	2	2	4
Contraseñas débiles	Ataque de diccionario	[PROVOCADO.1]	4	2	2	4
Cookies	Predicción de credenciales	[PROVOCADO.1]	3	2	2	3
Interesting Meta Tags Detected	Revelación no intencionada de información	[NO_INTENCIONADO.2]	2	1	1	1

Debe considerarse el uso de un segundo factor de autenticación, como por ejemplo el de envío de código de autorización por SMS o un número de autenticación por *token*, con el fin de detener los ataques de repetición.

Los puertos no utilizados deben ser cerrados. Además, se deben limitar las conexiones SSH al servidor, restringiendo a los equipos remotos por dirección IP y MAC.

Se recomienda desactivar la opción de autocompletamiento de campos en el código fuente del aplicativo. Además, deben evaluarse y limitarse el uso de los meta-tags y el tipo de información que se está entregando en los mismos.

Deberá establecerse el encabezado de respuesta HTTP X-Frame-Options, pues esto sirve para prevenir que la página pueda ser abierta en un frame, o iframe, lo cual evita los ataques de *clickjacking* sobre el sitio web.

Se deben establecer los parámetros de X-Frame de acuerdo con las necesidades del negocio. Los parámetros son: i) DENY, ii) SAMEORIGIN y iii) ALLOW-FROM URI.

Para evadir los ataques XSS, se sugiere incluir la siguiente cabecera: “header('X-XSS-Protection: 1;mode=block');”. También es importante establecer un conjunto de caracteres válidos para respuesta, como, por ejemplo: AddCharset utf-8 .html.

Asegurarse que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. En lo posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing.

Se debe implementar un IDS/IPS para el análisis de tráfico malicioso, esto permitirá alertar y bloquear los ataques de inundación, a los cuales el servidor es vulnerable.

4. Discusión

En una prueba de seguridad, el contar con una sola herramienta de análisis no es suficiente. Un *pentesting* efectivo requiere de la composición de estrategias y herramientas para cada una de las etapas que conforma un análisis. Los desarrolladores de OSS presentan interesantes soluciones al mercado, de todas ellas se han identificado como herramientas importantes para el propósito de este trabajo al distro Kali Linux (NMAP, Sparta, HPING3, Wireshark, Hydra, Metasploit Framework y Cookie Cadger), Mantra Linux (Nikto, OWASP (Wapiti, W3AF), APEX-SERT, Web Developer Plug In y VEGA.

Se recurrió a la metodología ECU@Risk (Crespo, 2017) para la identificación y valoración de amenazas y riesgos, al considerar además la herramienta de modelamiento de amenazas de Microsoft (Microsoft, 2017), lo cual ha facilitado priorizar las tretas a las cuales se debería aplicar una contramedida.

Las buenas prácticas de aseguramiento de plataformas y aplicaciones Web no recaen únicamente en la tecnología que se está utilizando. Muchas veces al pensar únicamente en la usabilidad, funcionalidad y estética, se dejan de lado importantes aspectos de seguridad; y cuando se incluyen diversos aspectos de seguridad, podría terminarse con una aplicación de uso muy complejo para el usuario. Por esta razón, una conclusión que puede rescatarse de (UNAM-CERT, 2016), es el mantener un equilibrio entre la funcionalidad y la seguridad.

Otra de las buenas prácticas es la de filtrar los datos. Como se había visto, un número de cédula 1212121212 fue aceptado. El filtrar los datos de entrada reduce el riesgo de contar con información errónea en la plataforma.

La importancia del cifrado de datos es inminente. Se pudo comprobar mediante el uso de un sniffer, y de acuerdo con (UNAM-CERT, 2016), cuando un atacante tiene la facilidad de realizar un ataque de hombre en el medio, se debe preocupar por la exposición de los datos durante el envío, transmisión y recepción de los mismos. Por tal razón, es necesario utilizar un protocolo de cifrado de información. Adicionando a esto, se ha visto que un sitio web es susceptible a ser descargado completamente para luego ser utilizado

en ataques de tipo phishing; por tal razón, la adopción de certificados digitales que validen, no solo el sitio web sino además la organización, es obligatoria.

Una sesión APEX genera identificadores de sesión aleatorios cuyas cookies expiran al momento de cerrar el navegador, lo cual reduce notablemente la posibilidad de ataques de fuerza bruta efectivos. Si bien estas *cookies* no pudieron ser identificadas con la herramienta Cookie Cadger, han logrado ser encontradas con el plugin Web Developer de Chrome.

Las herramientas VEGA, OWASP ZAP, han revelado el uso de los métodos GET, POST, PUT, DELETE provistos por un sistema web para el acceso a la información. De los anteriores, los dos últimos deben ser manejados con cautela, pues PUT permite cargar contenidos y DELETE eliminarlos. Estos métodos deben ser analizados y evaluados antes de su puesta en producción.

Las dos herramientas indicadas anteriormente también han permitido identificar las cabeceras HTTP expuestas. De estas, se puede indicar que “Anti-MIME-Sniffing header X-Content-Type-Options” aún no ha sido establecida a “no sniff”, permitiendo que navegadores antiguos interpreten el contenido de una forma diferente a la declarada originalmente.

5. Conclusiones y recomendaciones

La seguridad informática no se resume únicamente en la implementación de herramientas tecnológicas como *firewalls* y antivirus. Las contramedidas deben aportar a la mitigación de riesgos y amenazas de forma efectiva y deben ser parte de una adecuada cultura en el manejo de información, así como también técnicas de desarrollo de software, el lenguaje de programación que debe aplicarse y la configuración de la infraestructura computacional.

Según un artículo publicado por la UNAM, uno de los puntos más críticos de exposición a la web son los servidores, infraestructura tecnológica que no es visible para el usuario, pero que soportan toda la transaccionalidad y almacenamiento de datos producto de la interacción usuario – aplicación. De acuerdo con lo expresado por (UNAM-CERT, 2016), la mayoría de los problemas provocados no son necesariamente por fallas de los servidores o lenguajes utilizados, sino por malas prácticas de programación.

El ataque de fuerza bruta es otro factor que debe ser considerado en la protección del servidor. Como se había visto, el puerto SSH (utilizado para la administración remota) acepta conexiones con las cuales se pudo ejecutar un ataque de este tipo. Una contraseña simple puede ser fácilmente vulnerada, por cuanto se recomienda el uso de contraseñas complejas, las mismas que deben incluir caracteres alfabéticos en mayúsculas y minúsculas, caracteres numéricos y caracteres especiales.

Si bien una aplicación web puede estar bien desarrollada, la infraestructura computacional que apoya a sus actividades también puede verse afectada si las mismas no son aseguradas de la forma correcta. En las pruebas realizadas para este escenario, se ha podido ver que el servidor es susceptible a ataques de denegación de servicio. En estas pruebas, los ataques de inundación SYN Flood y TCP Flood resultaron exitosos, por lo tanto, el uso de un IDS/IPS es recomendado tal como lo sugieren (Mansoor, Muthuprasanna, & Vijay, 2006) y (Chakrabarti, Chakraborty, & Mukhopadhyay, 2010).

Bibliografía

Alcides, G. (2009). *Seguridad informática.* Antioquía, Colombia: Universidad de Antioquía.

Caballero, A. (2015). *Hacking con Kali Linux.* Lima.

Chakrabarti, S., Chakraborty, M., & Mukhopadhyay, I. (2010). Study of snort-based IDS. *ACM*, 43-47.

Crespo, E. (2017). ECU@Risk. *Metodología de Seguridad de la información para la gestión del riesgo informático aplicable a MPYMES*. Cuenca, Azuay, Ecuador.

Gallo, F. (2011). *Inseguridad Informática.* España.

Hernández, A. (2007). Fuzzing para pruebas de seguridad en software. *brainoverflow.org*.

Mansoor, A., Muthuprasanna, M., & Vijay, K. (2006). High Speed Pattern Matching for Network IDS/IPS. *IEEE Xplore*.

Microsoft. (2017). *SDL Threat Modeling Tool.* Obtenido de https://www.microsoft.com/en-us/sdl/adopt/threatmodeling.aspx

NMAP. (2017). *Resumen de Opciones*. Obtenido de https://nmap.org/man/es/man-briefoptions.html

Rocha, L. (2014). *Count Upon Security.* Obtenido de Incident Handling and Hacker Techniques: https://countuponsecurity.com/2014/05/29/simple-and-practical-attack-part-2/

Security, O. (2017). *Kali Tools*. Recuperado el 31 de 07 de 2017, de https://tools.kali.org/tools-listing

Spendolini, S. (2016). *Expert Oracle Application Express Security.* Apress.

Stallman, R. (2004). *Software libre para una sociedad libre.* Madrid: Traficantes de sueños.

The OWASP Project. (2017). *OWASP Testing Guide 4.0.* N/E: The OWASP Project.

UNAM-CERT. (2016). *Aspectos Básicos de la Seguridad en Aplicaciones Web.* Recuperado el 14 de 07 de 2017, de https://www.seguridad.unam.mx/historico/documento/index.html-id=17

University of Adelaide. (2015). *The Risk Management Handbook.* Sydney: Legal and risk.

Weidman, G. (2014). *Penetration Testing, A Hands-On Introduction to Hacking (1).* EEUU: No Starch Press.